(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 022 664 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
26.07.2000 Bulletin 2000/30

(51) Int. Cl.[7]: **G06F 17/30**

(21) Application number: **00300318.3**

(22) Date of filing: **18.01.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **21.01.1999 JP 1339499**

(71) Applicant:
International Business Machines Corporation
Armonk, NY 10504 (US)

(72) Inventors:
- **Kobayashi, Makoto**
  Winchester, Hampshire SO21 2JN (GB)
- **Shinozaki, Masahide**
  Winchester, Hampshire SO21 2JN (GB)
- **Sakairi, Takashi**
  Winchester, Hampshire SO21 2JN (GB)

(74) Representative:
Moss, Robert Douglas
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Method and system for sharing between browsers

(57) A method and a system for sharing a browser page wherein it is not necessary to install a controller for sharing on a computer in advance comprises a collaboration server for accumulating pages to be shared from a Web server retaining the original pages. The collaboration server comprises a caching manager that accumulates pages dynamically generated on an original Web server for sharing, a communication manager that controls sessions such as communication and participation/quitting among node managers controlling a browser on each user machine, an ordinary Web server for downloading facilities for implementing sharing of a Web server and an embedder that embeds in each page a page manager for controlling pages. The plural user machines to be shared comprise an existing Web browser capable of running Java and Script, a page manager embedded in each page. The page manager comprises two parts, namely, a page controller and a page communicator. The page controller detects changes in a page element, communicates them to another machine by way of a page communicator, and receives changes in a page of another computer and then reflects the same changes to its own page element. A page communicator manages communication between the Node manager and the Page controller. The node manager for controlling a browser is provided for each process and manages communication between each page manager and a server.

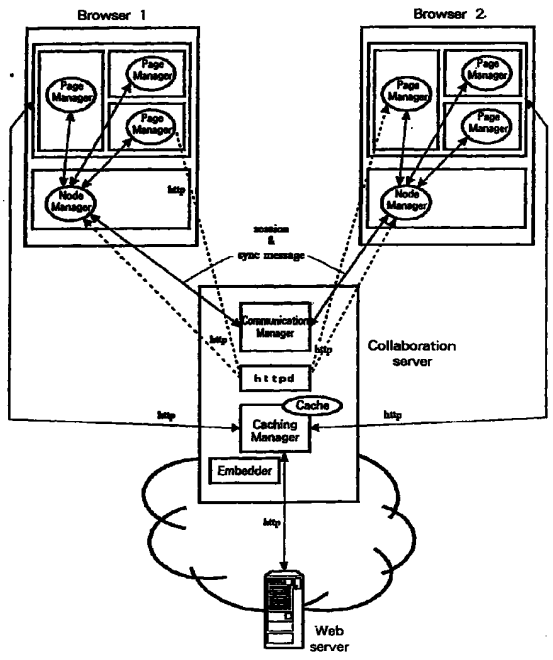

FIG. 2

EP 1 022 664 A2

Description

Technical Field of the Invention

[0001] The present invention relates to a method and a system for sharing between browsers, particularly to a method and a system for implementing high-performance and real-time sharing for an existing Web browser and an existing Web page, without a user having to install it, by embedding in the page itself a facility for controlling each element in the page.

Background of the Invention

[0002] Various methods are known for sharing a standard Web page in an as-is format by using an existing Web browser. For instance, European Patent Application EP 0833260 A2 is known. In such background art, however, it is necessary for every user to have a sharing system installed in advance. In EP 0833260 A2, synchronization of browser status is implemented by data exchange for sharing through the two interfaces, namely, an application interface (specifically, the functions of an information event of page loading, an inquiry about a current page and page setting) included in the browser and an interface at an operating system level (specifically, the functions of GET and PUT for a message queue which a window of a browser has). At this time, the application interface and message queue interface are accessible only from outside a browser application process, and therefore, since a module for implementing the sharing is outside the browser, it must be installed in advance and should not be automatically downloadable as an applet. There was also a problem of having to install it on each individual platform since it relies on a browser-running OS or a window system.

[0003] Fig. 1 shows the working of browser sharing in European Patent Application 0833260 A2. To share a page of a Web server on a collaboration server, it is necessary that a program for sharing (WebShare) other than a browser is installed in advance for customers and a call centre operator. This program for sharing allows sharing by hooking a browser API and an event. Also, such a program must be installed on each individual platform since it relies on a browser-running OS or a window system.

Disclosure of the Invention

[0004] An object of the present invention is to provide a method and a system for sharing between browsers wherein it is not necessary to install in advance a controller for sharing into a computer.

[0005] Another object is to provide a method and a system for sharing between browsers which do not depend on the functions of an OS or a window system and operate on various platforms.

[0006] A further object is to provide a method and a system for sharing between browsers which are capable of advanced sharing techniques (for instance, allowing a customer to enter in a field of a form but prohibiting moving to another page) required in an application.

[0007] A still further object is to provide a method and a system for real-time sharing between browsers which require less data volume and allows a good response even on a narrow band width as in an end user environment such as a home.

[0008] Accordingly, the present invention provides the following. First, a collaboration server is provided for accumulating pages to be shared from a Web server retaining original pages. The collaboration server (hereafter, merely a "server") comprises a caching manager that accumulates pages dynamically generated on an original Web server for sharing, a communication manager that controls sessions such as communication and participation/quitting among node managers controlling a browser on each user machine, an ordinary Web server for downloading facilities for implementing sharing of a Web server (a node manager and a page manager) and an embedder that embeds in each page a page manager for controlling pages.

[0009] The user machines requiring sharing ability comprise an existing Web browser capable of running Java and Script, and a page manager embedded in each page. The page manager comprises two parts, namely, a page controller and a page communicator. The page controller detects changes in a page element, communicates them to another machine by way of the page communicator, and receives changes in a page of another computer and then reflects the same changes to its own page element. Changes in a page element refer to page loading, changes in value of text and buttons which are elements of a form, changes in a scroll position of a page, and operation of a remote pointer, etc. The page communicator manages communication between a node manager and the page controller. A node manager for controlling a browser is provided for each process and manages communication between each page manager and a server. Moreover, there is nothing unique for collaboration in the hardware configuration of the above-mentioned server. A user machine and a server are only different in name, and it is no problem if each user machine and server consist of exactly the same hardware.

[0010] The present invention enables implementation of advanced real-time sharing of a browser among a number of users which could not be implemented by any background art. In addition, synchronization of page loading, synchronization of input operations of form elements, synchronization of scrolling operations, synchronization of remote pointers, and synchronization of annotations are possible, and a client machine only requires an existing browser comprising functions of Java and a script. Since it does not require any external program or a module plug-in to be installed, a browser sharing system which does not burden a user

with installation and requires little data traffic for synchronization is provided.

Brief Description of the Drawings

[0011]

Fig. 1 is a schematic diagram of a system showing an example of conventional browser sharing of pages;

Fig. 2 is a schematic diagram showing a system according to the present invention for the sharing of browser pages;

Fig. 3 shows further detail of a browser and node manager employed in the system of Figure 2;

Fig. 4 is a diagram showing an example of processing of page loading in a nested frame;

Fig. 5 is a flowchart showing the process for setting up a browser for page sharing by the method of the present invention;

Fig. 6 is a flowchart for showing operation of a browser for page sharing by the method of the present invention; and

Fig. 7 is a diagram showing an example of hardware configuration of a server and plural computers used in the present invention.

Detailed Description of the Invention

[0012] Fig. 2 shows a diagram of the entire configuration of the present invention. A collaboration server comprises a caching manager that accumulates pages dynamically generated on an original Web server for sharing, an ordinary Web server (httpd) for downloading facilities for implementing sharing of a Web server, a communication manager that controls sessions among Node managers on each user machine, and an embedder that embeds a page manager. The facilities for sharing between plural computers (user machines) comprise two components, namely, a module (node manager) for controlling each process of a browser (Web browser 1 or Web browser 2) and a module for controlling each page (page manager). A page manager monitors a state of each page element in a page, detects changes and exchanges information with a corresponding remote page manager so as to dynamically perform setting of each page element to be in the same state. Also, for synchronization of windows with a nested frame structure, a page manager checks a hierarchical structure of a frame (n-th position of n-th nest) and, with this as an ID, communicates with a corresponding page manager. This hierarchical structure information can be obtained on any browser without being limited by a facility of cross frame security. While there are two user machines in Fig. 2, it is possible to share a browser likewise with three or more machines.

[0013] The node manager controlling a browser performs communication (session and synchronization) between each page manager and a server. The node manager resides in a page independent from the shared Web window and which does not migrate, and controls communication between page managers dynamically generated/terminated for each page loading. It also controls information across pages such as history. A page manager and a node manager are embedded as Java applets which have an identical domain. Thus, regardless of the domain of the original page in which the page manager is embedded, data communication by shared memory is performed between a page manager and a node manager on any browser without being limited by a facility of cross frame security.

[0014] By configuring them in such a manner, sharing of a real-time Web browser becomes possible since a facility for page sharing can be embedded in an existing HTML page between an existing Web server and a browser without changing its original structure.

[0015] Operation of the browser sharing system is explained in detail below based on operation inside a browser in Fig. 3.

1. Start of a Node manager

[0016] A node manager is loaded into a new browser window as it is opened when a customer has opened a page which indicates the start of sharing or has pressed a start of sharing button. This window exists, if seen by a user, independently of a window to be shared and is not closed during a sharing session. In this window, a user interface for switching a remote pointer and a normal mode, etc. are displayed. A Java socket is placed between this node manager and a communication manager that is a component of the server so that the shared data may be exchanged among plural user machines (Java is a trademark of Sun Microsystems Inc.). In addition, shared memory is allocated as a class variable of a node manager applet. In this shared memory, a queue of messages identifying a Page manager is organised.

2. Embedding of a page manager in a shared page

[0017] The page manager comprises a page controller consisting of a script for controlling each page element and a page communicator made of an applet which communicates with the node manager through the message queue. These two modules are embedded by an embedder module of the server. As an example of actual code, the following code is embedded immediately after a 〈BODY〉 tag comes out while reading char-

acter strings in an HTML page:-

〈SCRIPT
SRC="http://collabsvr/pagemgr.js"〉〈/SCRIPT〉
〈APPLET
CODE="http://collabsvr/pagemgr.class"〉〈/APPLE
T〉

[0018] As the page communicator and the node manager have an identical domain, data can be exchanged avoiding a problem of security regardless of the domain of the page to be embedded. Thus, a multi frame page comprising plural domains can be shared.

3. Detecting changes in a page element in a shared page by a Page manager

[0019] The page controller detects local changes to the following elements and communicates them to a corresponding remote page controller so as to implement synchronization of pages.

3-1. Mechanism for synchronizing page loading

[0020] Changes in a page occur along with the following operations:-

(1) A click on a link in a page by user operation.

(2) Operation on a browser menu (selecting Back/Forward button and a bookmark) by user operation.

(3) Autoload by description in a page (designation by META syntax and description in an applet or a script).

[0021] Of these, (1) is synchronized, as in the detection of a changing event of a form element described below, by detecting and communicating a click event. In the cases of (2) and (3), since a page controller on a current page cannot detect a load request event, the page controller in a newly loaded page posts a new URL to another node. Here, Fig. 4 shows an example of processing page loading in a nested frame. Step 1 of Fig. 4 illustrates the initial state of browsers 1 and 2. Fig. 4 illustrates a case where the following initial state (synchronous state) switches to a final state (synchronous state) (a case where frame set S2 changes to frame set S3).

Initial state

[0022]

FRAMESET [S1]

FRAME [F1]
FRAMESET [S2]

FRAME [F2]
FRAME [F3]
FRAME [F4]

Final state

[0023]

FRAMESET [S1]

FRAME [F1]
FRAMESET [S3]

FRAME [F5]
FRAME [F6]

[0024] S1, S2 and S3 are abbreviations for frame sets 1, 2 and 3, and F1, F2, F3, F4, F5 and F6 are abbreviations for frames 1, 2, 3, 4, 5 and 6 respectively. Loading of a frame set is performed instead by a page manager included in a page of the first frame (in the case that it is further nested and the first element is also a frame set, it should go below one by one).

[0025] In step 2 of Fig. 4, as frame set 2 jumps to frame set 3, page managers of frames 2, 3 and 4 are first destroyed and related message queues are released.

[0026] In step 3 of Fig. 4, frames 5 and 6 are loaded. Related messages are placed in message queues, and the page manager of frame 5 sends a message of "Load frame set 3."

[0027] In step 4 of Fig. 4, the node manager of browser 2 distributes the messages to the message queue of an identical frame hierarchy, and the page manager of frame 2 loads the URL of frame set 2 so it returns to a synchronous state.

3-2. Form element changing events

[0028] As for synchronization of form elements such as a text field or a button, changes are detected by the following two methods depending on their properties:-

(1) Detecting a user-operated event with a hook of an event handler.

(2) As for an element which does not necessarily generate an event when changing, detecting an event through polling by a timer.

[0029] Explanation of the method of (2) is omitted since it is a background art. In the case of (1), since user operation to a form element such as a button generates a click event, it is possible to have so-called a hook of an event caused by a page controller setting its

own handler to an on-click event handler. In general, however, there are cases where a user-defined event handler is already described in an element such as a button in JavaScript or VisualBasic Script, and in such cases, the original handler is not processed by merely replacing the event handler by a page controller so that the existing page does not operate as originally described. So, the original handler is rewritten on page loading to add a line for jumping to the handler for hooking. In this way, it becomes possible for a page controller to hook an event without affecting the original handler.

3-3. Page scrolling

[0030] While the position can easily be detected by hooking an event generated on scrolling (onScroll), dragging scroll bar keeps generating events and causes too much traffic, so the position is detected through polling by a timer.

3-4. Operation of remote pointer

[0031] A remote pointer may be added to an arbitrary page, in the case of Internet Explorer of Microsoft Corp. for instance, by adding it as a DIV element as follows (it can also be done with the same technique in the case of Netscape Communicator of Netscape Communications Corp.):-

```
var str = '〈DIV style="position:absolute; overflow:none; width=50px height=50px〉〈IMG src="pointer.gif"〉〈/DIV〉';
document.body.insertAdjacentHTML("AfterBegin", str);
```

[0032] A pointer is moved by moving DIV to a corresponding position with a mouseMove event to a window.

[0033] Fig. 5 explains on a flowchart the necessary processing to permit page sharing between browsers. When a user has logged in on a page which indicates the start of sharing or has pressed a start of sharing button provided on a page, a process of sharing is started. In step 510, a new browser window is opened, and a node manager is loaded there. This node manager window exists independently of a shared browser window, and is not closed during a sharing session. In this node manager window, a user interface for switching a remote pointer and a normal mode, etc. may be displayed. In step 520, the communication manager and the node manager which are components of a collaboration server are connected by a Java socket. This socket is used to exchange information for sharing among nodes. In step 530, a page manager is embedded in an HTML page by an embedding module of the server. The page manager comprises a page controller consisting of scripts for controlling each page element and a page communicator consisting of applets for communicating with node managers through a message queue. Since a page communicator and a node manager are in an identical domain, data can be exchanged avoiding a problem of security regardless of the domain of an HTML page to be embedded. Because of this, it becomes possible to share a multi frame page comprising pages of plural domains. The above concludes processing until sharing of a browser becomes possible.

[0034] Fig. 6 explains on a flowchart a preferred process for sharing browser operation according to the present invention. In reality, however, due to constraints of describing it on a flowchart, when an event is generated and a handler set for the event is to be activated, a decision block is used instead. Also, where a timer is set for regular processing even in the case of a process not using a handler, a simple loop is used instead in the flowchart.

[0035] In step 610, a browser loads an HTML page in which a page manager is embedded. Next, in step 620, shared memory is assigned as a class variable of the node manager's applet. A message queue with a page manager in it is created. Also, a Page controller sets its own handler to an onclick event handler.

[0036] Furthermore, the same process is performed for other events as required. In step 630, a decision on termination is made. Sharing is terminated if a sharing termination button is pressed. In step 640, a decision on page loading is made. Page loading is decided by the current page controller detecting an event of a click in the case of a click in a page by a user. Loading generated by operation on a browser menu (back, forward, a bookmark) or description in a page (META syntax, an applet, a script), etc. is decided by the page controller in a newly loaded page referring to a location attribute of a browser window.

[0037] In step 642, a page loading process is performed. In the case of page loading by user operation on a browser menu or description in a page, a Page controller in a newly loaded page posts a new URL to another node. In step 644, a page is unloaded. The browser unloads the current page along with page loading. And in step 646, the page manager is terminated. In the process of unloading the current page, the page manager on this page is terminated. At this time, the message queue page manager entry is eliminated and this shared memory is released.

[0038] In step 650, a decision on the form is made. As for synchronization of form elements such as a text field or a button, it is decided by the two methods depending on their properties. A user-operated event is decided by hooking an event handler. Changes in an element which does not necessarily generate an event when changing are decided by regularly checking the value with a timer. In step 655, form processing is performed. If the user operation is of a kind to generate an event, the same process as the page loading process in step 642 is performed. As for changes in an element which does not necessarily generate an event when

changing, the value of the changed form element is sent to another node. Processing returns to step 630 after that.

[0039] In step 660, a decision on scrolling is made. Scrolling operation is decided by regularly detecting the position with a timer. In step 665, a scrolling process is performed. The position of a new scroll is sent to the other node. Processing returns to step 630 after that.

[0040] In step 670, a decision on a pointer is made. It is decided by a mouseMove event to a window. In step 685, a pointer process is performed. The position of the new pointer is sent to the other node. A pointer can be added to any HTML page as follows as a DIV element:-

```
var str = '〈DIV style="position:absolute; overflow:none;
width=50px height=50px〉〈IMG src="pointer.gif"〉〈/DIV〉';
document.body.insertAdjacentHTML("AfterBegin", str);
```

[0041] The pointer is moved by moving DIV to a position acquired from a mouseMove event. Processing returns to step 630 after that.

[0042] In step 680, a decision on receiving is made. A page communicator decides whether a message has been received from another node by checking a message queue. In step 685, a receiving process is performed. Depending on the contents of a received message, an appropriate process is performed.

[0043] Thus, when a message instructing page loading by user operation on a browser menu or description in a page is received, the received URL is set to the location of the window and the same HTML page is loaded. When a message changing value of a form element is received, the form element is changed as instructed. When a message changing a position of a scroll is received, the position of the scroll is changed as instructed. When a message changing a position of a pointer is received, the position of the pointer is changed as instructed.

[0044] Fig. 7 shows an example of an embodiment of the hardware configuration of a server and plural computers (user machines) used in the present invention. System 100 comprises central processing unit (CPU) 1 and memory 4. CPU 1 and memory 4 are connected via bus 2 with hard disk device 13 as an auxiliary storage (or drives for storage media such as CD-ROM 26 and DVD 32) via IDE controller 25. Likewise, CPU 1 and memory 4 are connected via bus 2 with hard disk device 30 as an auxiliary storage (or drives for storage media such as MO 28, CD-ROM 29 and DVD 31) via SCSI controller 27. Floppy disk drive 20 is connected with bus 2 via floppy disk controller 19.

[0045] A floppy disk inserted into floppy disk drive 20 contains code or data of a computer program for giving instructions to a CPU and so on in synergy with an operating system to implement the present invention. This code or data can be recorded on this floppy disk, etc., hard disk device 13 (or a storage media such as MO, CD-ROM and DVD) and ROM 14, which is executed by being loaded to memory 4. This computer program code can also be compressed or divided into two or more so as to be recorded on two or more media.

[0046] System 100 has further user interface hardware comprising pointing device (a mouse, a joystick, etc.) 7 or a keyboard 6 for entry and a display 12 for providing a user with visual data. It is also possible to connect with a printer via parallel port 16 or connect with a modem via serial port 15. This system 100 can be connected with a network via serial port 15 and a modem or communication adapter 18 (Ethernet or Token-ring card) etc. so as to communicate with other Web servers, other computers and so on. In addition, it is possible to connect a remote transmitter-receiver with serial port 15 or parallel port 16 so as to exchange data by means of an infrared ray or an electric wave.

[0047] Speaker 23 receives a speech signal which is D/A (digital/analog) converted by audio controller 21 via amplifier 22 and outputs it as speech. In addition, audio controller 21 makes it possible to A/D (analog/digital) convert speech data received from microphone 24 and capture into the system speech data outside the system.

[0048] Thus, for example, the server and plural computers in the present invention is implementable by a communication terminal with a communication facility including an ordinary personal computer (PC) or a workstation, a notebook PC, a palmtop PC, a network computer, various household electrical appliances with a built-in computer such as a TV set, a game console with a communication facility, a telephone, a fax, a portable telephone, a PHS, an electronic organiser or combination of these.

Claims

1. A system for sharing pages between browsers, the system having a server and plural client computers, each client computer having a browser for browsing pages, a page manager controlling said pages and a node manager for controlling communication between said page manager and said server, wherein said page manager comprises:

   means for detecting changes in one of its own pages and sending said changes to said node manager for sending said changes to said server; and

   means for receiving changes in a page of another client computer from said node manager, and reflecting said changes in one of its own pages.

2. A system according to claim 1 wherein said server

comprises:

a caching manager that accumulates pages;

a communication manager that controls sessions among said plural client computers; and

an embedder that embeds in each page a page manager for controlling pages.

3. A system according to either claim 1 or claim 2 wherein said page manager includes a page controller and a page communicator, said page

controller comprising: means for detecting changes in a page element and sending said changes to said node manager by way of said page communicator; and

means for receiving changes in a page of another computer from said node manager by way of said page communicator and reflecting the received changes to one of its own page elements.

4. A system according to claim 3 wherein said changes in a page element are changes in page loading, changes in a form element including text and buttons, changes in a scroll position of a page or operation of a remote pointer.

5. A system according to claim 1 wherein said page manager analyzes frame hierarchical structure of a page and communicates with a corresponding page manager based on this analysis result.

6. A system according to any preceding claim wherein said node manager resides in a page independent from the page to be shared browser and which does not migrate and controls communication between page managers dynamically generated or terminated along with page loading.

7. A system according to any preceding claim wherein said node manager controls page information including the transition history of a page.

8. A system according to any preceding claim wherein said page manager and said node manager are embedded as Java applets which have an identical domain and data communication between said page manager and said node manager is performed via shared memory.

9. A server for sharing a page between browsers on respective plural computers, comprising:

means for receiving from any one of said computers a signal for sharing said browser page;

means for sending to said one computer a node manager controlling said browser;

means for receiving from said one computer a request for viewing a page on said browser;

means for sending to said one computer, according to said request for viewing a page, a request page in which a page manager for controlling pages is embedded;

means for receiving page change information sent by said page manager via said node manager; and

means for sending said page change information to another of said computers.

10. A method for sharing a browser page between browsers on respective plural computers, comprising the steps of:

on activating said browser of a computer, loading a node manager on the computer from a server;

establishing communication between said server and said node manager, said node manager assigning shared memory;

on page viewing on said browser, embedding a page manager on a requested page on said server;

establishing communication between said node manager and said page manager via said shared memory; and

sending changes in a page being viewed to said node manager via said shared memory, and receiving changes in a page on another computer from said node manager via said shared memory and reflecting said changes to a page being viewed on said first mentioned computer.

11. A computer program for sharing a browser page between browsers on plural computers, said program comprising:

means for establishing communication with a server;

means for assigning shared memory;

means for issuing a page request to said server

for a page to be viewed on said browser;

means for receiving from said server a page in which a page manager controlling pages is embedded; and

means for sending to said server changes in a page received from said page manager via said shared memory and means for receiving changes in a page of another computer from said server and sending said changes to said page manager via said shared memory.

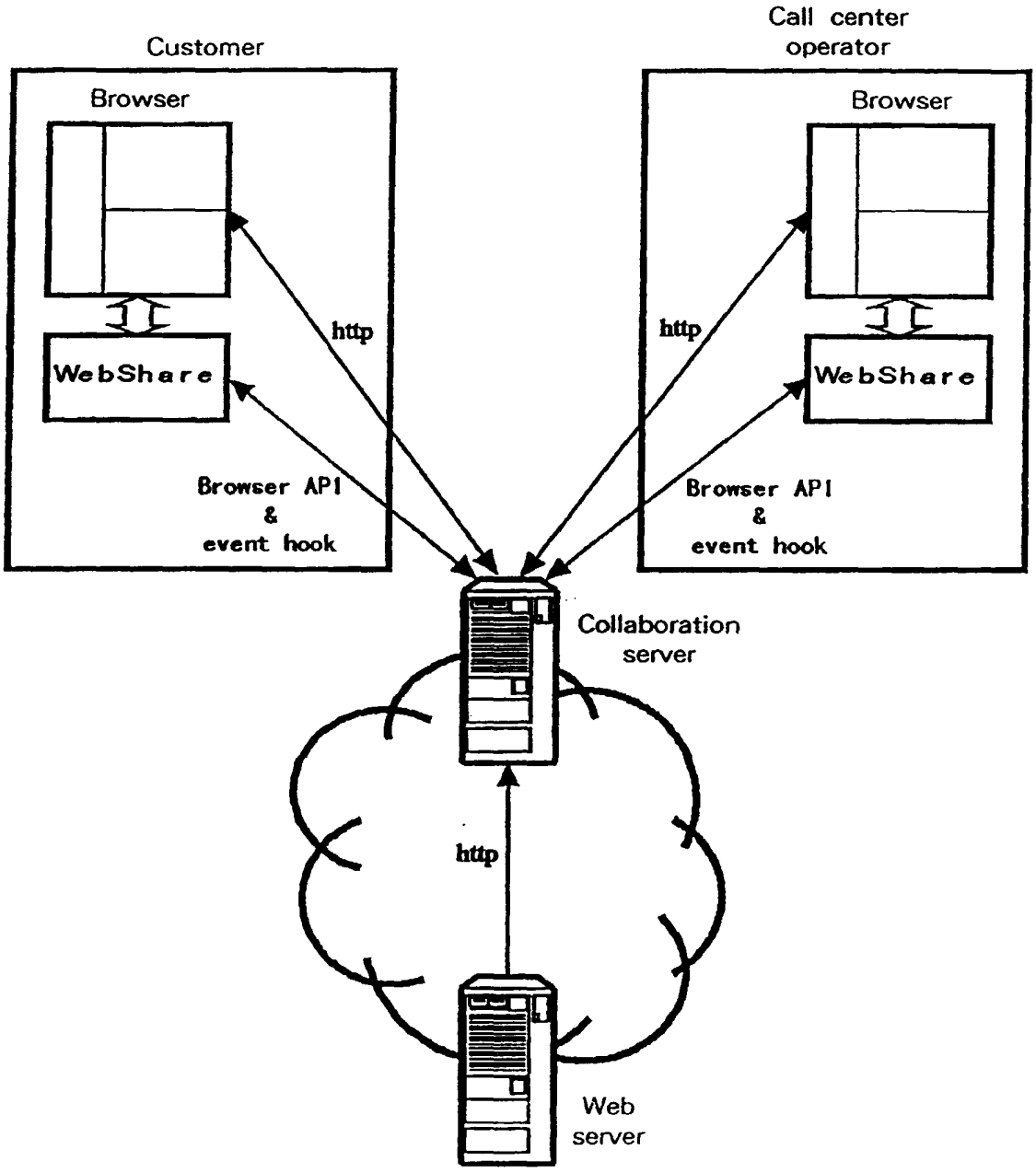
Customer
Browser
WebShare
http
Browser API
&
event hook
Call center
operator
Browser
WebShare
http
Browser API
&
event hook
Collaboration
server
http
Web
server

FIG. 1

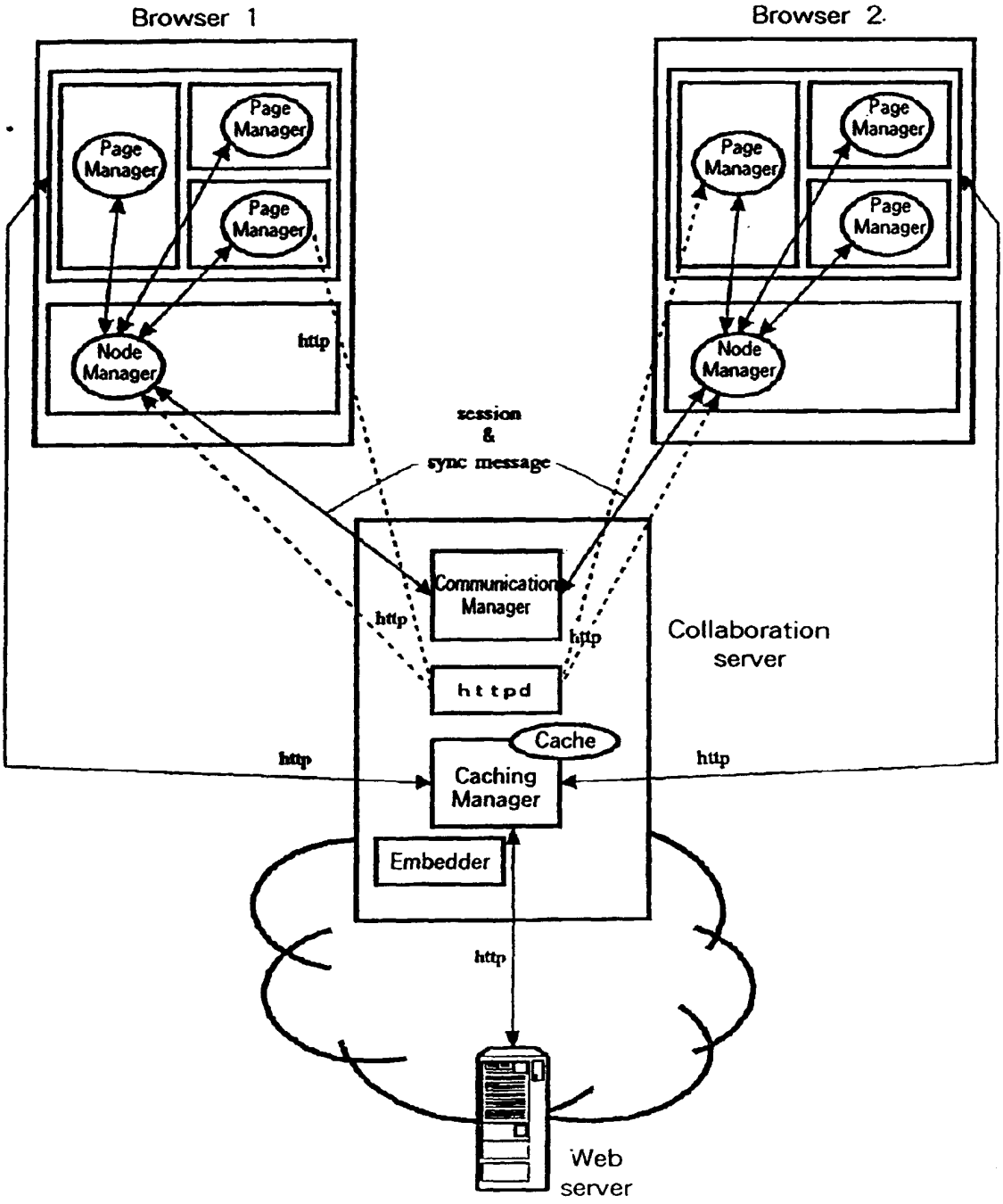
Browser 1
Browser 2
Page Manager
Page Manager
Page Manager
Node Manager
http
Page Manager
Page Manager
Page Manager
Node Manager
session
&
sync message
Communication Manager
http
http
Collaboration server
h t t p d
Cache
Caching Manager
http
http
Embedder
http
Web server

FIG. 2

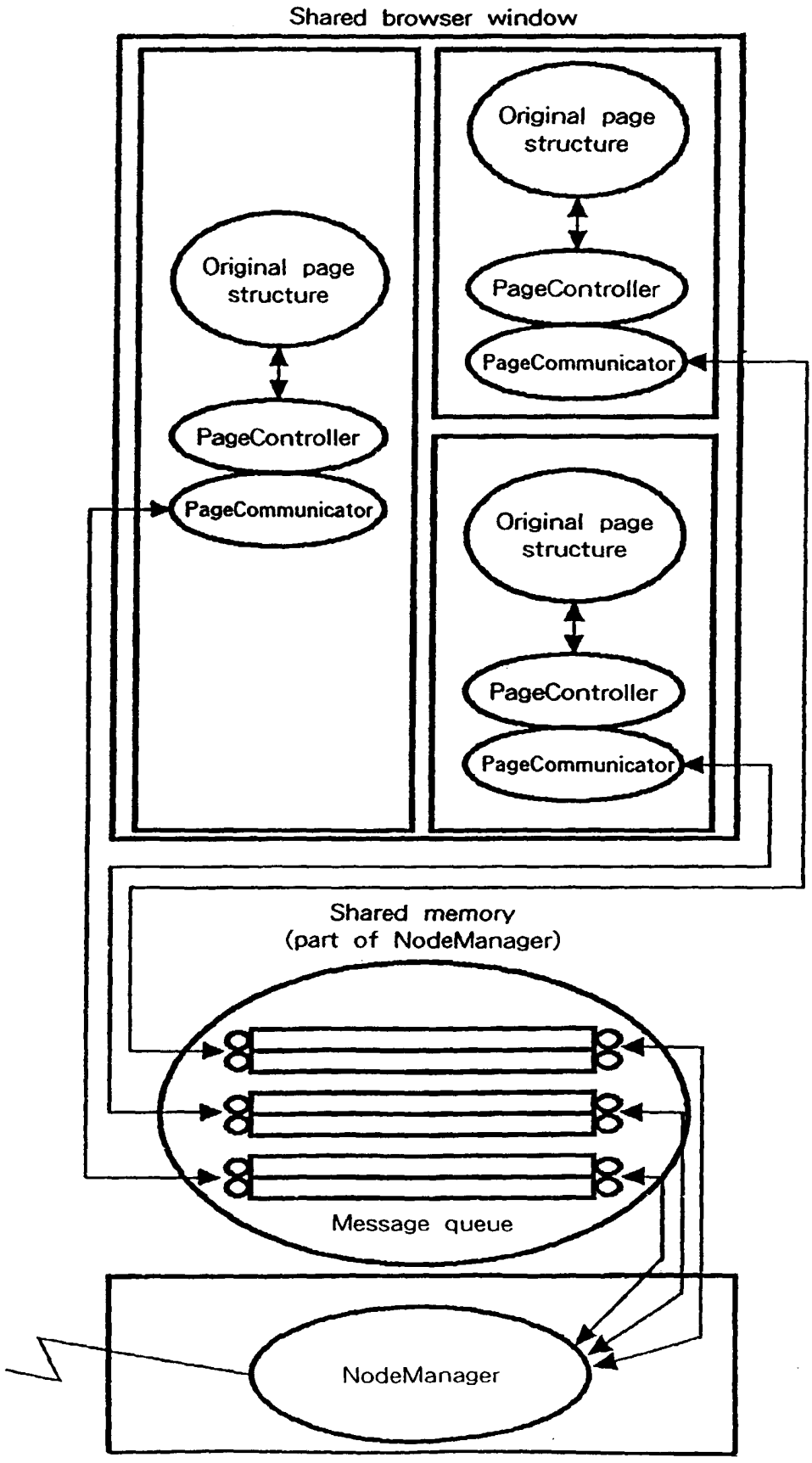
Shared browser window
Original page structure
PageController
PageCommunicator
Original page structure
PageController
PageCommunicator
Original page structure
PageController
PageCommunicator
Shared memory
(part of NodeManager)
Message queue
NodeManager

FIG. 3

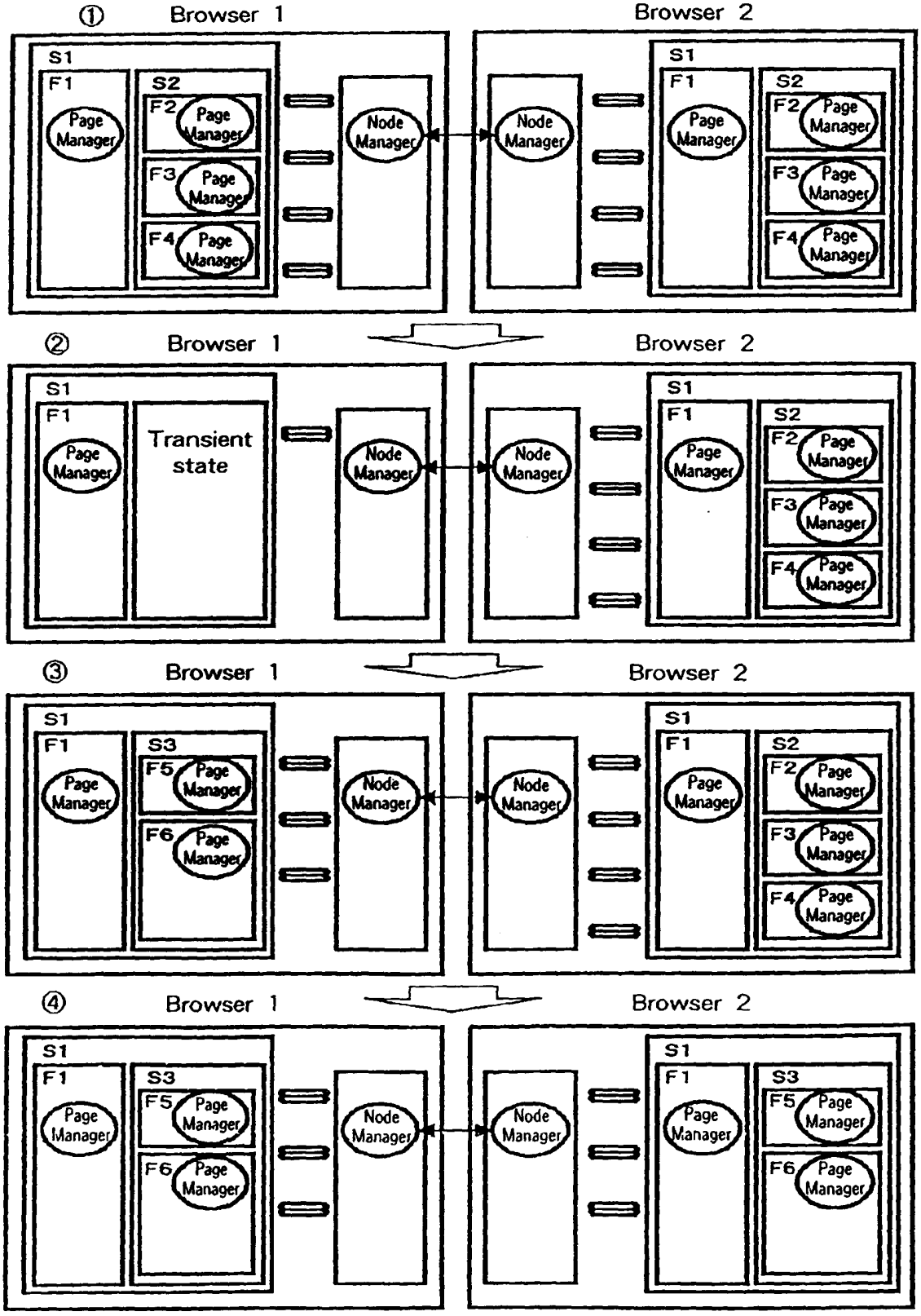
①
Browser 1
Browser 2
S1
F1
S2
F2
F3
F4
Page Manager
Node Manager
②
Transient state
③
S3
F5
F6
④

FIG. 4

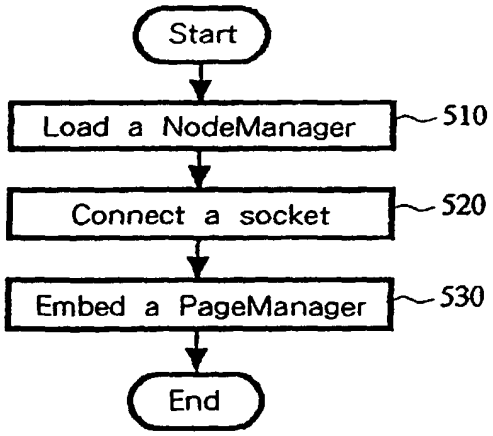
Start
Load a NodeManager
510
Connect a socket
520
Embed a PageManager
530
End

FIG. 5

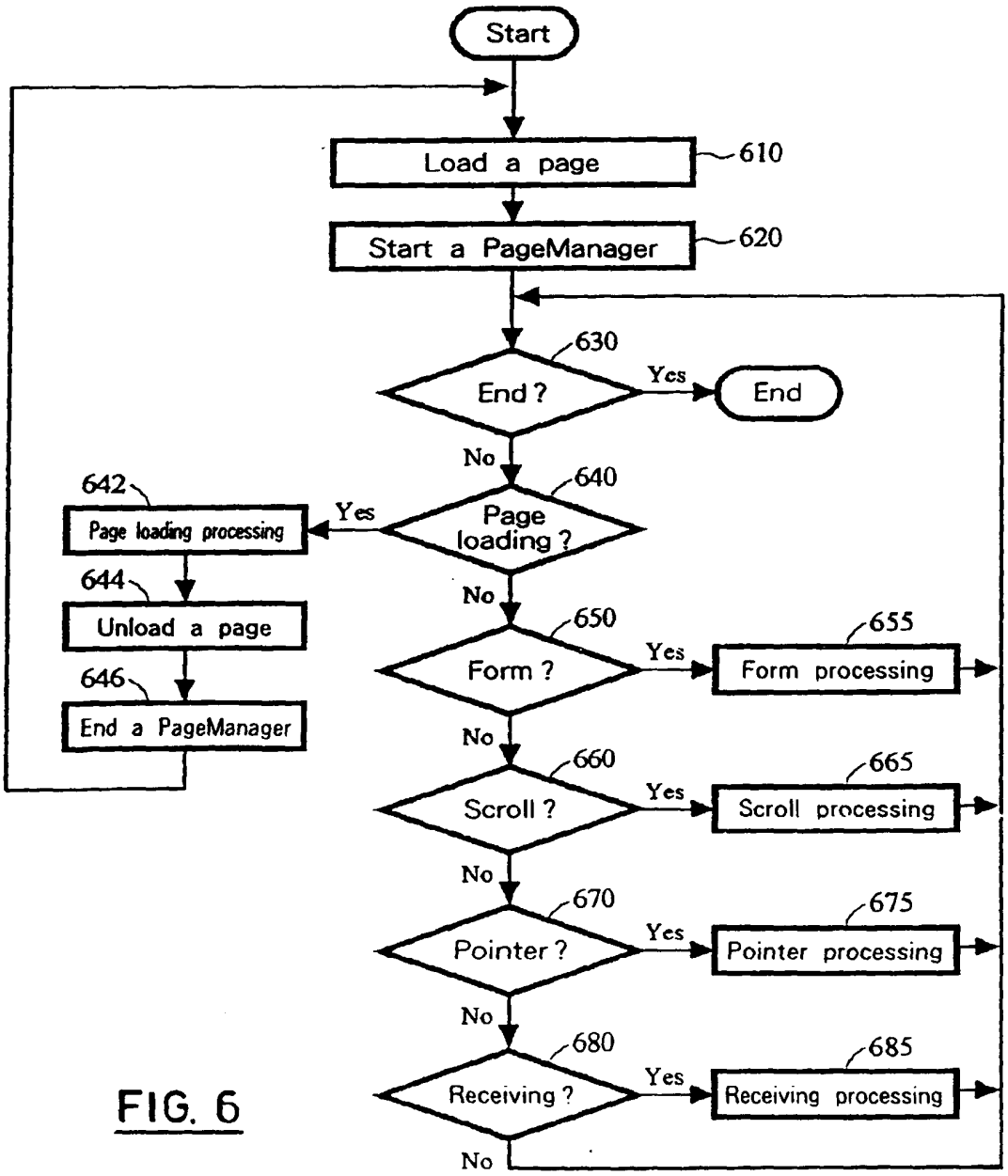
Start
Load a page
610
Start a PageManager
620
630
End ?
Yes
End
No
642
Page loading processing
Yes
640
Page loading ?
644
Unload a page
No
650
Form ?
Yes
655
Form processing
646
End a PageManager
No
660
Scroll ?
Yes
665
Scroll processing
No
670
Pointer ?
Yes
675
Pointer processing
No
680
Receiving ?
Yes
685
Receiving processing
No

FIG. 6

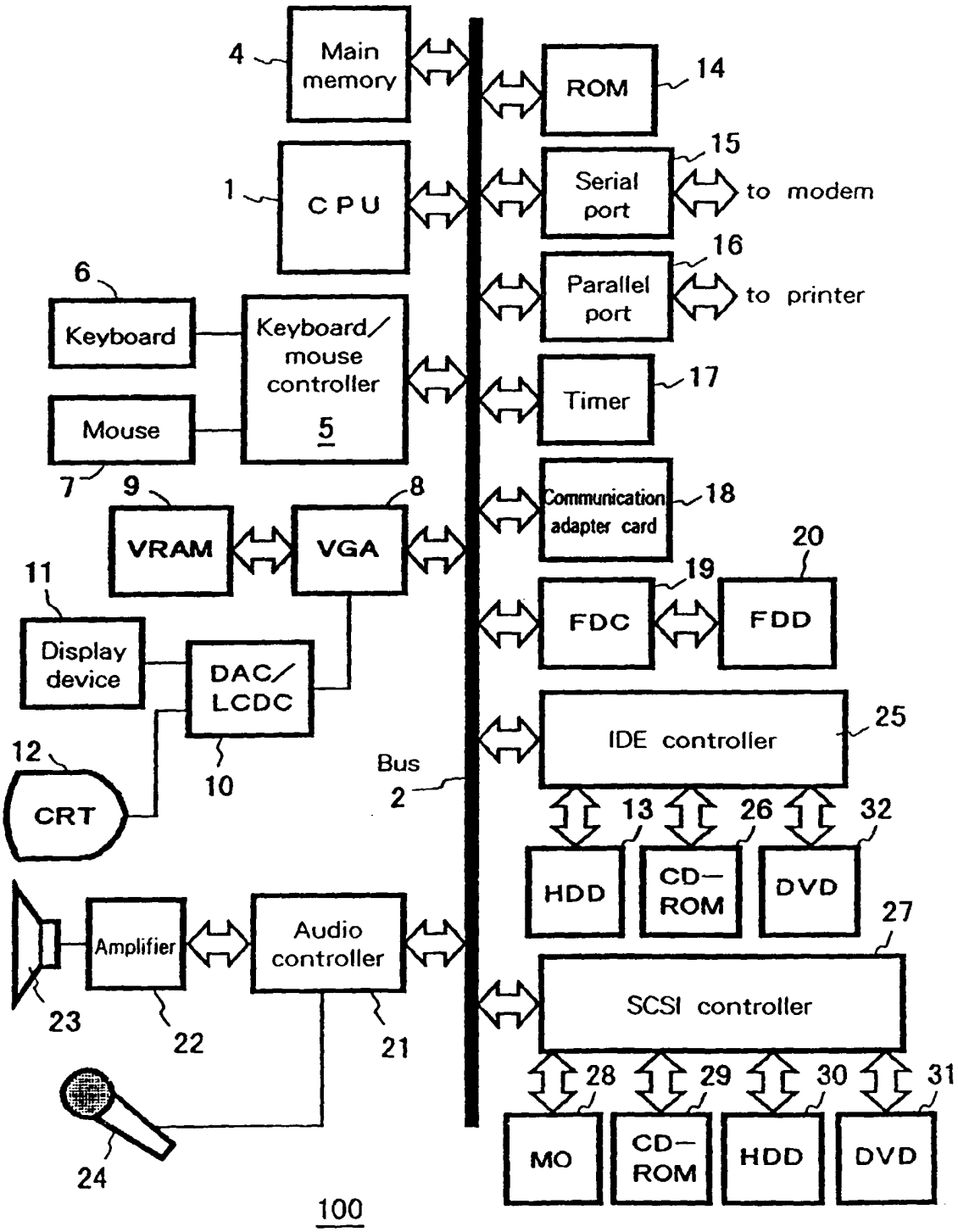
4
Main memory
ROM
14
15
1
CPU
Serial port
to modem
16
6
Parallel port
to printer
Keyboard
Keyboard/ mouse controller
5
Timer
17
Mouse
7
9
8
Communication adapter card
18
VRAM
VGA
20
11
19
Display device
FDC
FDD
DAC/ LCDC
25
IDE controller
12
10
Bus
2
CRT
13
26
32
HDD
CD-ROM
DVD
27
Amplifier
Audio controller
SCSI controller
23
22
21
28
29
30
31
MO
CD-ROM
HDD
DVD
24
100

FIG. 7